Basic GDB Usage

Combine the commands below to serve your debugging!

Invoke GDB / Quit

`$ gdb <Executable> [Command-line Args]` to invoke *gdb*

`(gdb) q` to quit

Set Breaks / Watch Variables

`b <function>` / `b <file>:<function>`

- break at function **entrance**, right after all arguments have been assigned
- `b` = `break`, only for short
- file name can precede with directories (e.g. `subdir/a.c`)

`b <line>` / `b <file>:<line>`

- break at line **begin**, printing the unexecuted line

`b ... if <cond>`

- conditional break only if `cond != 0` when reaches this breakpoint
- `cond` can be any valid expression

`rbreak <regex>`, `rbreak <file>:<regex>`

- match function names with `regex`
- see `man grep` for regular expressions supported

`watch <var>` / `watch <file>::<var>` / `watch <function>::<var>`

- break whenever `var` is modified
- `var` can be `*address` (e.g. `watch *0x600850`)
- watching requires special hardware support, so might not work under certain conditions

`rwatch ...`

- break whenever `var` is read

`awatch ...`

- break whenever `var` is read or written

`info b` / `info watch`

- view the infos of all breakpoints / watchpoints

`clear ...`

- clear breakpoint / watchpoint specified by `...` (just like how you set them)

`delete`

- delete all breakpoints and watchpoints

Run / Continue / Step Execution

`r`

- start running the program
- `r` = `run`, only for short

`c`

- continue running, **until** finish or reaching next breakpoint
- `c` = `continue`, only for short

`s` / `s <count>`

- step forward one / `count` instructions (**will go inside function calls**), then break again, printing the next unexecuted line
- `s` = `step`, only for short

`n` / `n <count>`

- step forward one / `count` instructions (**but will finish function calls in the line**)
- `n` = `next`, only for short

List Source Codes

`l`

- show 10 source lines around current location
- `l` = `list`, only for short

`l <line>` / `l <file>:<line>`

- show 10 source lines around a certain line

`l <function>`

- show 10 source lines around a function

`l -`

- show 10 source lines before previous listing

`l +`

- show 10 source lines after previous listing

Examine / Modify Data

`p <expr>`

- **execute** `expr` and show its **current** value
- `p` = `print`, only for short
- `expr` can be any valid expression at this point

`p <var=...>`

- set the value of `var`
- `var=...` is just an assignment expression in C, so this is only a special case of `p <expr>`

`p *<arr>@<len>`

- show the array `arr`'s first `len` contents

`p <arr>[<index>]@<len>`

- show the array `arr`s contents from `index` to `index + len - 1`

Examine Current Frame

`f`

- print out the current stack frame
- `f` = `frame`, only for short

`info f`

- print comprehensive informations of current stack frame

`info locals`

- print **current** value of all local variables

`info args`

- print **current** value of all function arguments

`p <local-var>@entry`

- show the variable `local-var`'s value **when enters the function**

Backtrace / Move among Frames

`bt`

- show back-tracing function call stack
- `bt` = `backtrace` = `where`, only for short
- from top -> bottom, order is later -> earlier

`bt full`

- show back tracing function call stack, with all local variable values

`f <frame-id>`

- go to specific stack frame numbered `id`
- can see the stacks' info through `bt`

`up <num>`

- go up `num` stack frames

`do <num>`

- go down `num` stack frames
- `do` = `down`, only for short

Misc

1. Run shell command by `!<shell-cmd>` / `shell <shell-cmd>`
2. Auto Completion by double tapping `TAB`